IE 498
Spring 2020
Homework 5
April 2, 2020

Name (Print): _____

You are required to show your work on each problem on this homework. The following rules apply:

- **Organize your work**, in a reasonably neat and coherent way, in the space provided. Work scattered all over the page without a clear ordering will receive very little credit.
- **Unsupported answers will not receive full credit**. A correct answer, unsupported by calculations, explanation, or mathematical work will receive no credit; an incorrect answer supported by substantially correct calculations and explanations might still receive partial credit.

Do not write in the table to the right.

Problem	Points	Score
1	25	
2	15	
3	15	
4	15	
5	15	
6	15	
Total:	100	

1. (25 points) **Backpropagation.**

   (a) Derive the backpropagation algorithm for a neural network with two hidden layers. (20 points)

$$\begin{aligned} Z^1 &= W^1 x + b^1, \\ H^1 &= \sigma(Z^1), \\ Z^2 &= W^2 H^1 + b^2, \\ H^2 &= \sigma(Z^2), \\ f(x;\theta) &= W^3 H^2 + b^3, \\ \rho\big(f(x;\theta), y\big) &= \big(y - f(x;\theta)\big)^2. \end{aligned}$$

   Each layer of the neural network has d_H hidden units, the input $x \in \mathbb{R}^d$, and the $y \in \mathbb{R}$. The parameters are $\theta = \{W^1, b^1, W^2, b^2, W^3, b^3\}$. The objective function is the squared error $\big(y - f(x;\theta)\big)^2$. Derive the backpropagation algorithm for calculating $\nabla_\theta \Big[\big(y - f(x;\theta)\big)^2\Big]$.

   *Note - State the dimensions of all variables and answers.

   i.) Calculate the gradient of the objective function $\rho\big(f(x;\theta), y\big)$ with respect to the output $f(x;\theta)$. That is, calculate $\frac{\partial \rho}{\partial f}$. Call this value δ^4.

   ii.) Calculate the gradient of the the objective function $\rho\big(f(x;\theta), y\big)$ with respect to the parameters of the third layer. That is, calculate $\frac{\partial \rho}{\partial W^3}$ and $\frac{\partial \rho}{\partial b^3}$. Write your answer in terms of δ^4 from part i.

   iii.) Calculate the gradient of the the objective function $\rho\big(f(x;\theta), y\big)$ with respect to the second layer output Z^2. That is, calculate $\frac{\partial \rho}{\partial Z^2}$. Write your answer in terms of δ^4 from part i. Call this δ^3.

   iv.) Calculate the gradient of the the objective function $\rho\big(f(x;\theta), y\big)$ with respect to the parameters of the second layer. That is, calculate $\frac{\partial \rho}{\partial W^2}$ and $\frac{\partial \rho}{\partial b^2}$. Write your answer in terms of δ^3 from part iii.

   v.) Calculate the gradient of the the objective function $\rho\big(f(x;\theta), y\big)$ with respect to the first layer output Z^1. That is, calculate $\frac{\partial \rho}{\partial Z^1}$. Write your answer in terms of δ^3 from part iii. Call this δ^2.

   vi.) Calculate the gradient of the the objective function $\rho\big(f(x;\theta), y\big)$ with respect to the parameters of the first layer. That is, calculate $\frac{\partial \rho}{\partial W^1}$ and $\frac{\partial \rho}{\partial b^1}$. Write your answer in terms of δ^2 from part v.

   (b) Suppose we have a dataset $(x^i, y^i)_{i=1}^M$. Write the stochastic gradient descent algorithm (with mini-batch size of 1) for training the network in part (a). (5 points)

2. (15 points) **Stochastic gradient descent.**

   (a) Suppose we randomly initialize a neural network $f(x;\theta)$ and train it for a long time with the final parameter estimate being θ^1. Suppose we again randomly initialize the same neural network model $f(x;\theta)$ and train it again for a long time with the final parameter estimate being θ^2. Will θ^1 and θ^2 always be similar?

   (b) Let $Y = f(X)$. The universal approximation theorem states that, for every $\epsilon > 0$, there exists a neural network $f(x;\theta)$ and a parameter choice θ^* such that

   $$\mathbb{E}_{X,Y}\left[\, \|Y - f(X;\theta^*)\| \,\right] < \epsilon. \tag{1}$$

   If we estimate θ using stochastic gradient descent, why might it not converge to the θ^* satisfying (1)?

   (c) What is the vanishing gradient problem? What are two methods or model architectures which can help address this challenge?

   (d) Write the softmax function for a K dimensional input.

   (e) In order to mathematically show that stochastic gradient descent converges, what conditions does the learning rate α_k have to satisfy? What is an example of a specific learning rate function which satisfies these conditions?

   (f) Describe, mathematically if possible, two methods for regularization.

3. (15 points) **Training algorithms**.
   (a) What is the RMSProp algorithm?
   (b) What is the advantage of RMSprop over standard stochastic gradient descent?
   (c) What is the deep Q-learning algorithm?
   (d) What is a standard algorithm to select the actions in the deep Q-learning algorithm?

4. (15 points) **Initializations and normalizations.**

   (a) Write the batch normalization algorithm (during training).

   (b) What is layer normalization?

   (c) What is the Xavier initialization?

   (d) Construct an example where the neural network will not train (i.e., the gradient with respect to *at least one* of the parameters is always 0). In other words, find a neural network architecture and a choice of initial parameters such that the gradient with respect to *at least one* of the parameters is always 0 while training with gradient descent. It is sufficient to consider mean-squared error, a single data sample (x, y), and a single hidden layer with a single ReLU unit.

   (e) We separately train M neural network models $f(x;\theta^1), \dots f(x;\theta^M)$ using stochastic gradient descent. Furthermore, each time we train a new model the initial parameter is randomly initialized. Therefore, the trained models $\theta^1, \dots, \theta^M$ are i.i.d. RVs. Is it mathematically correct to use the following ensemble model?

$$g(x) = f\left(x; \frac{1}{M}\sum_{m=1}^{M} \theta^m\right)$$

   Why or why not?

5. (15 points) **Convolution networks.**

   (a) Mathematically describe the convolution operation taking into account the stride, padding, number of input channels and number of output channels.

   (b) Why is the convolution operation useful?

   (c) Describe the effects and use of the hyperparameters described in (a).

   (d) Consider a convolution operation with 1 input channel, C output channels, stride of 1, padding is 0, and filter size 3×3. The convolution is performed on an image with size 30×30. What are the dimensions of the output of the convolution operation on this image?

   (e) Mathematically describe a 3D convolution. Consider a 3-D image with C_{in} input channels (i.e., $I \in \mathbb{R}^{C_{in} \times d_x \times d_y \times d_z}$). Include multiple output channels in the convolution.

6. (15 points) **PyTorch and Implementation Questions.**

   (a) What would be an advantage of Float16 versus Float32?

   (b) How does computational cost (defined as the number of arithmetic operations) grow with the number of mini-batch samples M? How does the GPU memory required grow with the number of mini-batch samples M?

   (c) Suppose the data set is too large to fit into CPU memory. How would you train a model?

   (d) Suppose we do not include "optimizer.step()" in our PyTorch code. What happens?

   (e) What does "optimizer.zero_grad()" do?

   (f) What PyTorch command moves a tensor to the GPU?

   (g) Suppose we are training a model using synchronized gradient descent on N machines. On each machine, the mini-batch gradient with respect to the model parameters is the tensor g. In order to take a synchronized gradient descent step, these mini-batch gradients need to be averaged across the entire cluster. What PyTorch command averages these tensors from all of the machines?

   (h) Suppose we are training an LSTM network on a data sequence $(X_t, Y_t)_{t=1}^{\infty}$. Suppose that every τ time steps we calculate the gradient and update the model. However, we never truncate the backpropagation through time. How does the computational cost of a parameter update grow with time t (i.e., what is the order of magnitude of the computational cost to calculate the gradient at time t)?

   (i) Suppose we now use truncated backpropagation through time (with truncation length τ) for training the LSTM. What order of magnitude is the computational cost?